یادگیری عمیق

جلسه ۱۲

مبانی یادگیری ماشینی (۲)

Fundamentals of Machine Learning (2)

کاظم فولادی قلعه

دانشکده مهندسی، پردیس فارابی

دانشگاه تهران

http://courses.fouladi.ir/deep

یادگیری عمیق

مبانی یادگیری ماشینی

۱

یادگیری ماشینی

صورت‌های یادگیری ماشینی

FORMS OF MACHINE LEARNING

صورت‌های یادگیری	
یادگیری استقرائی *Inductive Learning*	**یادگیری استنباطی** *Deductive Learning*
یادگیری جزء به کل	**یادگیری کل به جزء**
یادگیری یک تابع یا قاعده‌ی عمومی (درست/نادرست) از روی جفت‌های خاص ورودی ـ خروجی / مثال‌ها	**یادگیری تحلیلی** *Analytical Learning*
	حرکت از یک قاعده‌ی عمومی شناخته‌شده به قاعده‌ی جدیدی که منطقاً استلزام می‌شود. (مفید است، زیرا امکان پردازش کارآمدتر را فراهم می‌کند.)

صورت‌های یادگیری ماشینی

یادگیری استقرائی

یادگیری استقرائی
Inductive Learning

یادگیری جزء به کل

یادگیری یک تابع یا قاعده‌ی عمومی (درست/نادرست)
از روی جفت‌های خاص ورودی ـ خروجی / مثال‌ها

یادگیری بانظارت *Supervised*	یادگیری بی‌نظارت *Unsupervised*	یادگیری تقویتی *Reinforcement*	یادگیری نیمه‌نظارتی *Semisupervised*	یادگیری خودنظارتی *Self-Supervised*
یادگیری نگاشت ورودی به خروجی با دیدن مثال‌های برچسب‌دار	یادگیری الگوهای درون ورودی بدون وجود فیدبک صریح (مثل clustering)	یادگیری از روی یک سری تقویت‌ها (پاداش‌ها و جریمه‌ها)	یادگیری با وجود تعداد کمی مثال برچسب‌دار و مجموعه‌ی بزرگی از داده‌های بی‌برچسب	یادگیری بانظارت بدون استفاده از برچسب‌های تهیه شده توسط انسان / برچسب‌ها از روی داده‌های ورودی تولید می‌شوند.

تقسیم‌بندی بر اساس **نوع فیدبک** موجود برای یادگیری

سه نوع اصلی یادگیری

شاخه‌های یادگیری ماشینی

یادگیری بانظارت

یادگیری بانظارت، رایج‌ترین شکل یادگیری ماشینی است:
یادگیری نگاشت داده‌های ورودی به تارگت‌های معلوم
بر اساس مجموعه‌ای از مثال‌های داده شده

عموماً، تقریباً تمامی کاربردهای درخشان یادگیری عمیق، از نوع یادگیری بانظارت هستند.

مانند:
بازشناسی نوری حروف، بازشناسی گفتار، طبقه‌بندی تصاویر، ترجمه‌ی زبان

شاخه‌های یادگیری ماشینی

یادگیری بانظارت

SUPERVISED LEARNING

یادگیری بانظارت
Supervised Learning

یک **مجموعه‌ی آموزشی** از N جفت ورودی–خروجی نمونه داده شده است:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

که در آن هر y_j به‌وسیله‌ی یک تابع مجهول f تولید شده است:

$$y = f(x)$$

یک تابع h را کشف کنید که تابع واقعی f را تقریب بزند.

شاخه‌های یادگیری ماشینی

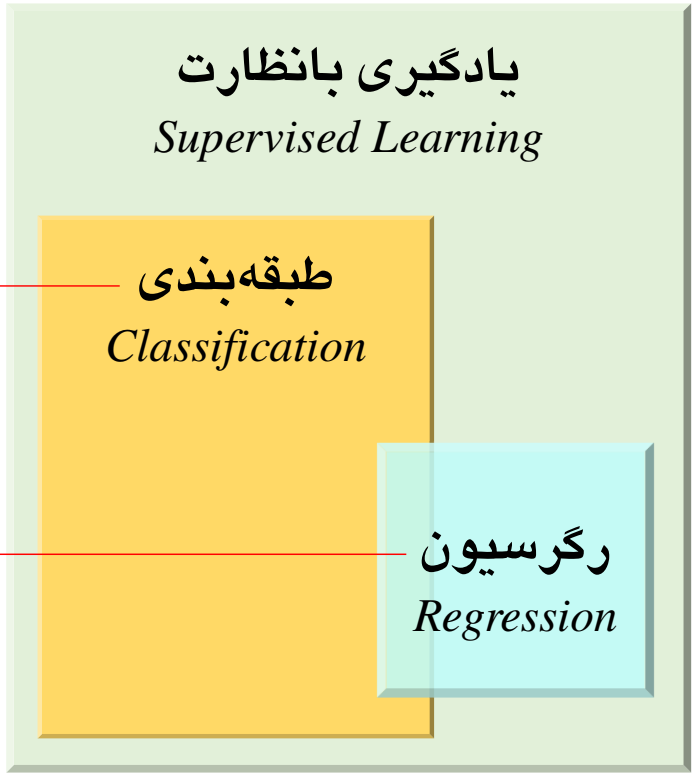
یادگیری بانظارت: مفاهیم کلیدی

یادگیری بانظارت *Supervised Learning*	
مجموعه‌ی آموزشی *Training Set*	مجموعه‌ی جفت‌های ورودی–خروجی معلوم برای آموزش یادگیرنده
مجموعه‌ی آزمایشی *Test Set*	مجموعه‌ی جفت‌های ورودی–خروجی معلوم برای آزمایش یادگیرنده
هدف / تارگت *Target*	تابع f که باید یاد گرفته شود
فرضیه *Hypothesis*	تابع مجهول h که باید تقریب مناسبی برای f باشد
فضای فرضیه *Hypothesis Space*	مجموعه‌ی همه‌ی توابع منتخب h برای تقریب f

$\mathcal{H}$

شاخه‌های یادگیری ماشینی

دو دسته‌ی مهم از مسائل یادگیری بانظارت : طبقه‌بندی و رگرسیون



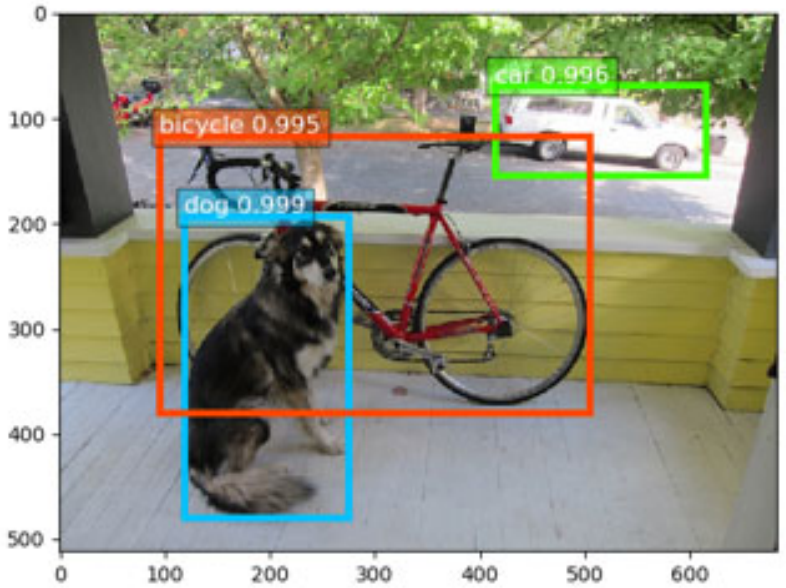
وقتی خروجی y از یک مجموعه‌ی متناهی مشخص انتخاب شود.

وقتی خروجی y یک عدد باشد (یافتن امید شرطی یا متوسط y)

شاخه‌های یادگیری ماشینی

نمونه‌هایی از کاربردهای یادگیری بانظارت

تولید دنباله *Sequence Generation*	**پیش‌بینی یک شرح برای توصیف یک تصویر داده شده** تولید دنباله اغلب می‌تواند به‌صورت یک سری از مسائل طبقه‌بندی فرمول‌بندی شود.
پیش‌بینی درخت نحو *Syntax Tree Prediction*	**پیش‌بینی تجزیه‌ی یک جمله‌ی داده شده به درخت نحو آن**
آشکارسازی اشیا *Object Detection*	**رسم یک چهارگوش دور اشیای خاص داخل یک تصویر داده شده** آشکارسازی اشیا می‌تواند به‌صورت یک مسئله‌ی طبقه‌بندی بیان شود (چهارگوش‌های کاندیدای متفاوت، طبقه‌بندی محتوای هر یک) یا مسئله‌ی طبقه‌بندی و رگرسیون توأم (رگرسیون برداری برای تعیین مختصات گوشه‌ها)
بخش‌بندی تصویر *Image Segmentation*	**رسم یک ماسک در سطح پیکسل بر روی یک شیئ خاص در یک تصویر داده شده**



آشکارسازی اشیا
Object Detection

بخش‌بندی تصویر
Image Segmentation

شاخه‌های یادگیری ماشینی

یادگیری بدون نظارت

UNSUPERVISED LEARNING

در یادگیری بدون نظارت،
یادگیری حاوی یافتن تبدیل‌های مفید داده‌های ورودی بدون کمک گرفتن از هرگونه تارگت است.

اهداف: بصری‌سازی داده‌ها، فشرده‌سازی داده‌ها، نویززدایی از داده‌ها، درک بهتر از همبستگی موجود در داده‌ها

یادگیری بی‌نظارت، معمولاً یک گام لازم برای فهم بهتر یک مجموعه داده
پیش از تلاش برای حل یک مسئله‌ی یادگیری با نظارت است.

کاهش ابعاد (dimensionality reduction) و **خوشه‌بندی** (clustering)
دو مسئله‌ی معروف در یادگیری بی‌نظارت هستند.

شاخه‌های یادگیری ماشینی

یادگیری خود نظارتی

SELF-SUPERVISED LEARNING

یادگیری خود نظارتی،
نوع خاصی از یادگیری با نظارت است که در آن از برچسب‌های تولید شده توسط انسان استفاده نمی‌شود.

در این نوع از یادگیری، برچسب وجود دارد،
اما برچسب‌ها از روی داده‌های ورودی و معمولاً با استفاده از یک الگوریتم هیوریستیک تولید می‌شوند.

برای مثال: خودکدگذارها (autoencoders). نمونه‌ی معروفی از یادگیری خودنظارتی هستند
که در آنها تارگت‌های تولید شده همان ورودی‌ها بدون هیچ تغییری هستند.

پیش‌بینی فریم بعدی در یک ویدئو با داشتن فریم‌های قبلی،
پیش‌بینی کلمه‌ی بعدی در یک متن با داشتن کلمات قبلی
یادگیری با نظارت زمانی (temporally supervised learning): داده‌های نظارت از آینده‌ی داده‌های ورودی می‌آیند.

شاخه‌های یادگیری ماشینی

یادگیری تقویتی

REINFORCEMENT LEARNING

یادگیری تقویتی،
عامل اطلاعاتی در مورد محیط خود دریافت می‌کند و
یاد می‌گیرد بهترین کنش‌هایی را انتخاب کند که پاداش او را حداکثر می‌سازد.

برای مثال:
یک شبکه‌ی عصبی که به صفحه‌ی یک بازی ویدئویی نگاه می‌کند و کنش‌های بازی را در خروجی تحویل می‌دهد،
به‌گونه‌ای که امتیاز آن حداکثر شود، می‌تواند با یادگیری تقویتی آموزش داده شود.

در حال حاضر یادگیری تقویتی بیشتر یک زمینه‌ی پژوهشی است و خارج از زمینه‌ی بازی‌ها
(مانند یادگیری بازی‌های آتاری [Google DeepMind] یا بازی Go)
موفقیت عملی چشمگیری نداشته است.

انتظار می‌رود موفقیت یادگیری تقویتی در حوزه‌ی گسترده‌ای از کاربردهای دنیای واقعی دیده شود، مانند:
خودروهای خودران، رباتیک، مدیریت منابع، تربیت و ...

یادگیری عمیق

مبانی یادگیری ماشینی

۲

ارزیابی مدل‌های یادگیری ماشینی

ارزیابی مدل‌های یادگیری ماشینی

EVALUATING MACHINE-LEARNING MODELS

هدف در یادگیری ماشینی، دستیابی به مدل‌هایی است که **قابلیت تعمیم (generalization)** داشته باشند
[یعنی بتوانند بر روی داده‌هایی که هرگز آنها را ندیده‌اند، به‌خوبی عمل کنند]
و **بیش‌برازش (overfitting)** مانع اصلی در قبال این موضوع است.

از آنجا که ما تنها می‌توانیم چیزی را کنترل کنیم که بتوانیم آن را مشاهده کنیم،
ضروری است که قادر باشیم قدرت تعمیم یک مدل را به‌طور مطمئن اندازه‌گیری کنیم.

⇓

چگونگی اندازه‌گیری تعمیم ⇐ چگونگی ارزیابی یک مدل یادگیری ماشین

ارزیابی مدل‌های یادگیری ماشینی

مجموعه‌های آموزشی، اعتبارسنجی، آزمایشی

TRAINING, VALIDATION, AND TEST SETS

مجموعه‌ی داده‌ها را به سه زیرمجموعه تقسیم می‌کنیم:

فرآیند آموزش روی **داده‌های آموزشی** انجام می‌شود.
مدل بر اساس **داده‌های اعتبارسنجی** ارزیابی می‌شود.
پس از آماده شدن مدل، برای آخرین بار روی **داده‌های آزمایشی** مورد آزمون و بررسی قرار می‌گیرد.

ارزیابی مدل‌های یادگیری ماشینی

چرا فقط از دو مجموعه‌ی آموزشی و آزمایشی استفاده نکنیم؟

چرا فقط از دو مجموعه‌ی آموزشی و آزمایشی استفاده نکنیم؟

زیرا:
توسعه‌ی یک مدل همیشه حاوی **تنظیم پیکربندی** آن است،
(مانند: انتخاب تعداد لایه‌ها، اندازه‌ی لایه‌ها، ... [هایپرپارامترها: hyperparameters])

تنظیم این پارامترها با استفاده از یک سیگنال فیدبک کارآیی شبکه روی داده‌های اعتبارسنجی انجام می‌شود.
خود این تنظیم نوعی یادگیری است که با استفاده از داده‌های اعتبارسنجی انجام شده است.

نشت اطلاعاتی
Information Leaks

هرگاه هایپرپارامتری از مدل را بر اساس کارآیی آن روی مجموعه‌ی اعتبارسنجی تنظیم می‌کنیم، مقداری اطلاعات در مورد داده‌های اعتبارسنجی به مدل نشت می‌کند.

بنابراین باید از یک مجموعه داده‌ی کاملاً متفاوت که قبلاً هرگز مشاهده نشده است برای آزمایش مدل استفاده کنیم:
مجموعه‌ی آزمایشی.
مدل نباید به هیچ گونه اطلاعاتی در مورد این مجموعه‌ی آزمایشی دسترسی داشته باشد (حتی به‌طور غیرمستقیم).

ارزیابی مدل‌های یادگیری ماشینی

روش‌های اعتبارسنجی

تقسیم داده‌ها به مجموعه‌های آموزشی، اعتبارسنجی، آزمایشی به‌نظر سرراست می‌رسد،
اما روش‌های پیشرفته‌ای وجود دارد که زمانی که داده‌های کمی موجود است مفید واقع می‌شوند.

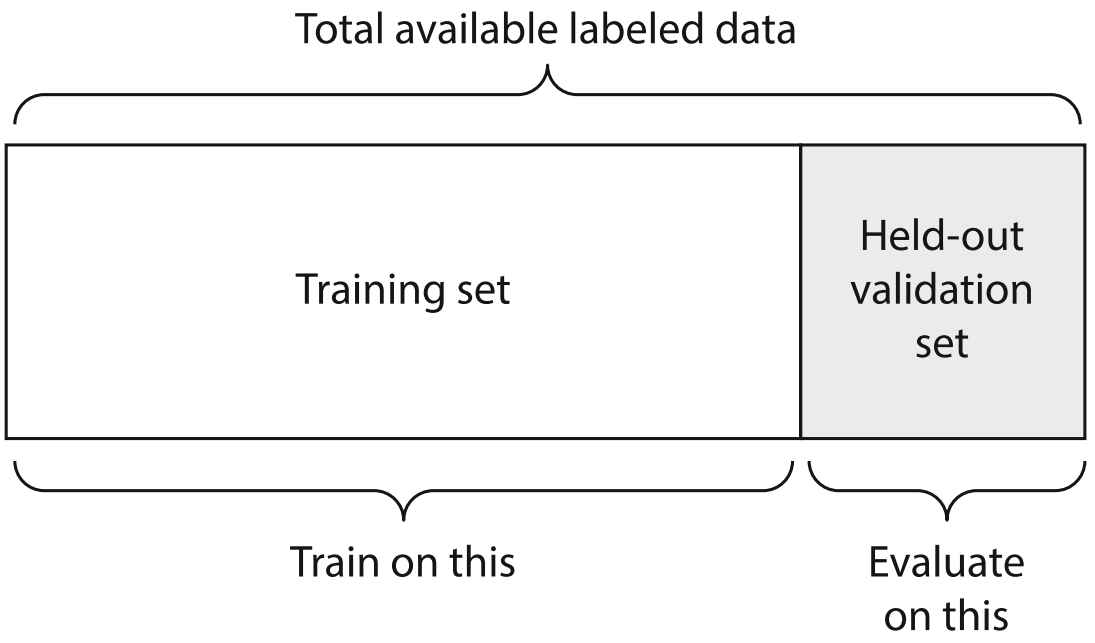
- **اعتبارسنجی ساده نگه‌داشت–برون‌گذاشت**
  Simple Hold-Out Validation
- **اعتبارسنجی *K-fold***
  K-fold Validation
- **اعتبارسنجی *K-fold* تکراری با بُر زدن**
  Iterated K-fold Validation with Shuffling

ارزیابی مدل‌های یادگیری ماشینی

اعتبارسنجی ساده نگه‌داشت–برون‌گذاشت

SIMPLE HOLD-OUT VALIDATION

بخشی از داده‌های آموزشی را به‌منظور اعتبارسنجی جدا می‌کنیم.



این ساده‌ترین پروتکل ارزیابی است و از یک نقیصه رنج می‌برد:
اگر داده‌های اندکی موجود باشد، آن‌گاه مجموعه‌های اعتبارسنجی و آزمایشی ممکن است حاوی تعداد نمونه‌های بسیار کمی برای نمایندگی آماری داده‌ها باشند.

ارزیابی مدل‌های یادگیری ماشینی

اعتبارسنجی ساده نگه‌داشت–برون‌گذاشت

Hold-out validation

```
num_validation_samples = 10000

np.random.shuffle(data)

validation_data = data[:num_validation_samples]
data = data[num_validation_samples:]

training_data = data[:]

model = get_model()
model.train(training_data)
validation_score = model.evaluate(validation_data)

# At this point you can tune your model,
# retrain it, evaluate it, tune it again...

model = get_model()
model.train(np.concatenate(
          [training_data,validation_data]))

test_score = model.evaluate(test_data)
```

Shuffling the data is usually appropriate.

Defines the validation set

Defines the training set

Trains a model on the training data, and evaluates it on the validation data

Once you've tuned your hyperparameters, it's common to train your final model from scratch on all non-test data available.

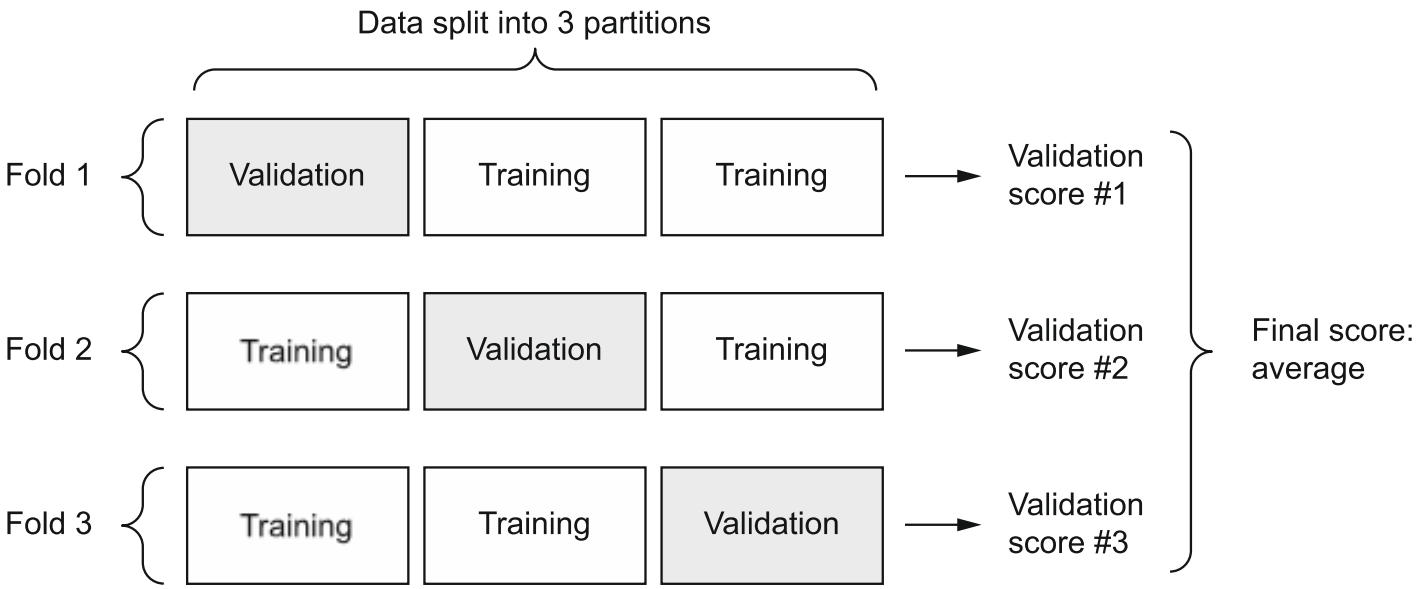
ارزیابی مدل‌های یادگیری ماشینی

اعتبارسنجی K-fold

K-FOLD VALIDATION

داده‌های موجود را به K سلول با اندازه‌ی مساوی افراز می‌کنیم (معمولاً $K = 4,5$)،
برای هر سلول i، یک مدل را بر روی $K-1$ سلول باقیمانده آموزش می‌دهیم
و آن را روی سلول i ارزیابی می‌کنیم.

امتیاز نهایی اعتبارسنجی مدل = متوسط K امتیاز اعتبارسنجی حاصل.



این روش زمانی کمک کننده است که کارآیی مدل ما واریانس بالایی بر اساس تقسیم آموزش–آزمایش داشته باشد.
این روش مانند روش ساده، ما را از به‌کارگیری یک مجموعه‌ی اعتبارسنجی مجزا برای کالیبراسیون مدل معاف نمی‌کند.

ارزیابی مدل‌های یادگیری ماشینی

اعتبارسنجی K-fold

K-fold cross validation

```
k = 4
num_validation_samples = len(data) // k

np.random.shuffle(data)

validation_scores = []

for fold in range(k):
    validation_data = data[num_validation_samples * fold:
       num_validation_samples * (fold + 1)]
    training_data = data[:num_validation_samples * fold] +
       data[num_validation_samples * (fold + 1):]

    model = get_model()
    model.train(training_data)
    validation_score = model.evaluate(validation_data)
    validation_scores.append(validation_score)


validation_score = np.average(validation_scores)


model = get_model()
model.train(data)
test_score = model.evaluate(test_data)
```

Selects the validation data partition

Uses the remainder of the data as training data. Note that the + operator is list concatenation, not summation.

Creates a brand-new instance of the model (untrained)

Validation score: average of the validation scores of the k folds

Trains the final model on all non-test data available

ارزیابی مدل‌های یادگیری ماشینی

اعتبارسنجی K-fold تکراری با بُر زدن

ITERATED K-FOLD VALIDATION WITH SHUFFLING

استفاده از اعتبارسنجی K-fold به‌تعداد P مرتبه
و بُر زدن (shuffling) داده‌ها در هر مرتبه پیش از تقسیم آن به K قسمت.

امتیاز نهایی = متوسط امتیازهای کسب شده در هر مرتبه از اجرای اعتبارسنجی K-fold

در پایان تعداد $P \times K$ مدل تحت آموزش و ارزیابی قرار گرفته است،
که می‌تواند بسیار پرهزینه باشد.

این روش مختص موقعیت‌هایی است داده‌های اندکی در اختیار داریم
و نیاز داریم مدل با حداکثر دقت ممکن ارزیابی شود.

ارزیابی مدل‌های یادگیری ماشینی

نکات مهم

نمایندگی داده‌ها
Data Representativeness

هم مجموعه‌ی آموزشی و هم مجموعه‌ی آزمایشی باید نماینده‌ی همه‌ی داده‌های موجود باشند (از همه‌ی کلاس‌ها به تعداد کافی در هر دو موجود باشد).

⇐ لازم است پیش از تقسیم داده‌ها بین مجموعه‌های آموزش و آزمایش آنها را به‌صورت تصادفی بر بزنیم (randomly shuffle).

پیکان زمان
The Arrow of Time

اگر هدف حل یک مسئله‌ی پیش‌بینی آینده بر اساس گذشته است، نباید پیش از تقسیم داده‌ها آنها را بر بزنیم و ترتیب آنها تصادفاً تغییر دهیم (این کار موجب نشت زمانی (temporal leak) می‌شود و باعث می‌شود مدل بر روی داده‌هایی از آینده آموزش ببیند).

در چنین مواردی باید مطمئن بود که همه‌ی داده‌های آزمایشی، از نظر زمانی پس از داده‌های آموزشی قرار داشته باشند.

افزونگی در داده‌ها
Redundancy in the Data

اگر برخی نقاط داده‌ای چند بار ظاهر شده باشند، بر زدن آنها و تقسیم آنها به مجموعه‌های آموزشی و اعتبارسنجی، موجب بروز افزونگی بین این دو مجموعه می‌شود. در نتیجه، تست روی بخشی از داده‌های آموزش انجام می‌شود (بدترین کار ممکن).

باید مطمئن باشیم که مجموعه‌های آموزشی و اعتبارسنجی مجزا باشند.

یادگیری عمیق

جلسه ۱۲

مبانی یادگیری ماشینی (۲)

Fundamentals of Machine Learning (2)

کاظم فولادی قلعه

دانشکده مهندسی، پردیس فارابی

دانشگاه تهران

http://courses.fouladi.ir/deep

یادگیری عمیق

مبانی یادگیری ماشینی

۳

پیش‌پردازش داده‌ها، مهندسی ویژگی‌ها، یادگیری ویژگی‌ها

پیش‌پردازش داده‌ها، مهندسی ویژگی‌ها، یادگیری ویژگی‌ها

DATA PREPROCESSING, FEATURE ENGINEERING, AND FEATURE LEARNING

چگونه باید داده‌های ورودی و تارگت‌ها را برای خوراندن به شبکه‌های عصبی آماده کنیم؟

بسیاری از تکنیک‌های **پیش‌پردازش داده‌ها** و **مهندسی ویژگی‌ها**، مختص به یک حوزه‌ی خاص هستند
(مانند: داده‌های متنی یا داده‌های تصویری، ...)

در اینجا، پایه‌های مشترک بین تمامی حوزه‌های داده‌ای بررسی می‌شوند.

پیش‌پردازش داده‌ها برای شبکه‌های عصبی

DATA PREPROCESSING FOR NEURAL NETWORKS

هدف از پیش‌پردازش داده‌ها، مناسب‌تر سازی داده‌های موجود برای شبکه‌های عصبی است.

- **برداری‌سازی**
  Vectorization
- **نرمال‌سازی**
  Normalization
- **برخورد با مقادیر گم‌شده**
  Handling Missing Values
- **استخراج ویژگی**
  Feature Extraction

پیش‌پردازش داده‌ها برای شبکه‌های عصبی

برداری‌سازی

VECTORIZATION

برداری‌سازی
Vectorization

تمام ورودی‌ها و تارگت‌ها در شبکه‌های عصبی باید به
تانسورهایی از داده‌های ممیز شناور (یا در موارد خاص اعداد صحیح)
تبدیل شوند.

مانند: کدگذاری تک–داغ برای کلمات و ...

پیش‌پردازش داده‌ها برای شبکه‌های عصبی

نرمال‌سازی

NORMALIZATION

نرمال‌سازی
Normalization

به‌طور کلی، خوراندن داده‌هایی که مقادیر نسبتاً بزرگی دارند یا داده‌های ناهمگن به یک شبکه‌ی عصبی، کار چندان ایمنی نیست.
(انجام این کار می‌تواند موجب تحریک به‌روزرسانی گرادیانی بزرگی شود که از همگرا شدن شبکه جلوگیری می‌کند.)

ویژگی‌های ضروری داده‌ها برای یادگیری ساده تر توسط شبکه‌های عصبی

- **مقادیر کوچک** (small values):
  اکثر داده‌ها باید در بازه‌ی [0,1] باشند.
- **مقادیر همگن** (homogenous):
  همه‌ی ویژگی‌ها باید مقادیری در بازه‌های تقریباً مشابه داشته باشند.

روش‌های زیر برای نرمال‌سازی قوی‌تر (استانداردسازی) در عمل متداول است، اما همیشه ضروری نیست:

- **نرمال‌سازی هر ویژگی به‌طور مستقل** تا میانگین آن صفر شود.
- **نرمال‌سازی هر ویژگی به‌طور مستقل** تا انحراف استاندارد آن یک شود.

```
x -= x.mean(axis=0)
x /= x.std(axis=0)
```

نرمال‌سازی داده‌ها

تکنیک‌ها

DATA NORMALIZATION: TECHNIQUES

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

بازمقیاس‌دهی (نرمال‌سازی *Min-Max*)
Rescaling (Min-Max Normalization)

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

نرمال‌سازی با میانگین
Mean Normalization

$$x' = \frac{x - \text{average}(x)}{\text{std}(x)}$$

استاندارد سازی
Standardization

پیش‌پردازش داده‌ها برای شبکه‌های عصبی

برخورد با مقادیر گم‌شده

HANDLING MISSING VALUES

برخورد با مقادیر گم‌شده
Handling Missing Values

در برخی موارد ممکن است مقادیر برخی از ویژگی‌ها در برخی از نمونه‌ها موجود نباشد
(مقدار گم‌شده).

در شبکه‌های عصبی، روش ایمن این است که
مقدار داده‌های ورودی ناموجود را برابر با **صفر** قرار دهیم.
به‌شرط آنکه **صفر** یک مقدار معنادار نباشد.

شبکه هنگام مواجهه با داده‌ای که مقدار آن برابر با صفر است،
یاد می‌گیرد که صفر به‌معنی داده‌ی ناموجود است و شروع به نادیده گرفتن آنها می‌کند.

نکته: اگر انتظار می‌رود مقادیر گم‌شده در داده‌های آزمایشی وجود داشته باشد،
اما شبکه با داده‌هایی فاقد هرگونه مقدار گم‌شده آموزش دیده باشد،
آن‌گاه این شبکه روش نادیده گرفتن مقادیر گم‌شده را نخواهد آموخت!
در این موارد باید به‌صورت مصنوعی نمونه‌های آموزشی دارای درایه‌های گم‌شده را تولید کنیم:
چند نمونه‌ی آموزشی را چند بار کپی می‌کنیم و چند ویژگی که در برخی داده‌های آزمایشی وجود ندارد را حذف می‌کنیم.

پیش‌پردازش داده‌ها برای شبکه‌های عصبی

مهندسی ویژگی‌ها

FEATURE ENGINEERING

استخراج ویژگی
Feature Extraction

در مهندسی ویژگی‌ها،
از دانایی موجود در مورد داده‌ها و الگوریتم یادگیری ماشینی مورد نظر (در اینجا شبکه‌های عصبی) استفاده می‌شود و تبدیل‌هایی بر روی داده‌ها پیش از ورود به مدل از طریق کدنویسی اعمال می‌شود تا موجب بهبود عملکرد الگوریتم شود.

در بسیاری از موارد، منطقی نیست که توقع داشته باشیم یک مدل یادگیری ماشینی قادر باشد از **داده‌های کاملاً دلخواه** یاد بگیرد.

داده‌ها باید به‌شیوه‌ای به مدل ارائه شوند که کار مدل را ساده‌تر کنند.

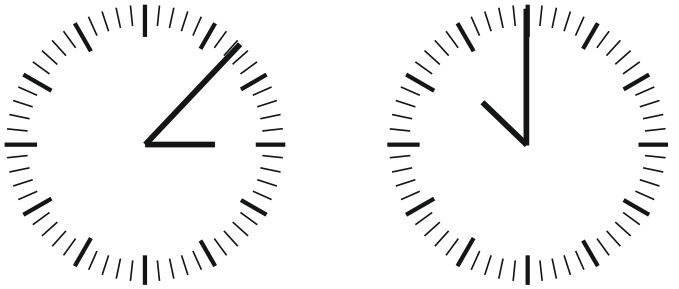
پیش‌پردازش داده‌ها برای شبکه‌های عصبی

مهندسی ویژگی: مثال (خواندن زمان از روی یک ساعت)

فرض می‌کنیم می‌خواهیم مدلی توسعه بدهیم که
تصویر یک ساعت را به‌عنوان **ورودی** دریافت کند و زمان روز را به‌عنوان **خروجی** تحویل بدهد.

اگر پیکسل‌های خام تصویر را به‌عنوان ورودی در نظر بگیریم، یا یک مسئله‌ی دشوار یادگیری ماشینی سروکار خواهیم داشت.
(نیاز به CNN با منابع محاسباتی بالا و داده‌های آموزشی بسیار زیاد)

Raw data: pixel grid



اگر این مسئله را در سطح بالا درک کنیم (روش خواندن زمان از روی ساعت توسط انسان‌ها) می‌توانیم ویژگی‌های ورودی بهتری برای الگوریتم یادگیری بیابیم (مثل: مختصات نوک عقربه‌ها)

Better features: clock hands' coordinates	{x1: 0.7, y1: 0.7} {x2: 0.5, y2: 0.0}	{x1: 0.0, y2: 1.0} {x2: -0.38, 2: 0.32}

حتی می‌توانیم از مختصات قطبی استفاده کنیم که در این‌صورت ویژگی‌های لازم، زاویه‌ی هر عقربه است. در اینجا که به‌حدی ساده می‌شود که دیگر حتی نیازی به یادگیری ماشینی نیست!

Even better features: angles of clock hands	theta1: 45 theta2: 0	theta1: 90 theta2: 140

پیش‌پردازش داده‌ها برای شبکه‌های عصبی

مهندسی ویژگی: ضرورت

اساس مهندسی ویژگی‌ها: ساده‌ترسازی یک مسئله از طریق بیان آن به‌صورتی ساده‌تر

پیش از یادگیری عمیق، مهندسی ویژگی‌ها بسیار حیاتی بود،
زیرا مدل‌های کم‌عمق فضاهای فرضیه‌ی به‌اندازه‌ی کافی غنی ندارند تا ویژگی‌های مفید را خودشان یادبگیرند؛
خوشبختانه، یادگیری عمیق مدرن نیاز به مهندسی را تا حد بسیاری حذف کرده است:
زیرا این مدل‌ها قادر به استخراج خودکار ویژگی‌های سودمند از داده‌های خام هستند.

اما این به‌آن معنا نیست که در استفاده از شبکه‌های عصبی عمیق به مهندسی ویژگی‌ها نیازی نداریم.

دلایل نیاز به مهندسی ویژگی‌ها در یادگیری عمیق

- **امکان حل مسائل به‌صورتی بهتر با استفاده از منابع کمتر با ویژگی‌های خوب**
- **امکان حل مسائل با حجم داده‌ی بسیار کمتر**

یادگیری عمیق

مبانی یادگیری ماشینی

۴

بیش‌برازش و کم‌برازش

بیش‌برازش و کم‌برازش

OVERFITTING AND UNDERFITTING

بیش‌برازش در هر مسئله‌ی یادگیری ماشینی اتفاق می‌افتد.
درک چگونگی رویارویی با بیش‌بردازش برای تسلط بر یادگیری ماشینی ضروری است.

مسئله‌ی بنیادین در یادگیری ماشینی، کشمکش میان **بهینه‌سازی (سازگاری)** و **تعمیم** است.

هدف این بازی رسیدن به تعمیم خوب است،
اما ما تعمیم را کنترل نمی‌کنیم، بلکه ما فقط می‌توانیم مدل را بر اساس داده‌های آموزشی تنظیم کنیم.

در ابتدای آموزش، بهینه‌سازی و تعمیم با یکدیگر همبسته هستند:
هر چه اتلاف روی داده‌های آموزشی کمتر باشد، روی داده‌های آزمایشی هم کمتر است (وضعیت **کم‌برازش**).
اما پس از چند تکرار مشخص روی داده‌های آموزشی، بهبود تعمیم متوقف می‌شود (شروع **بیش‌برازش**).
مدل شروع می‌کند به یادگیری الگوهایی که مختص داده‌های آموزشی است (آنها را حفظ می‌کند)؛
اما هنگامی که داده‌های جدید مطرح می‌شوند، این الگوها گمراه‌کننده یا نامربوط هستند.

سازگاری و تعمیم

سازگاری *Consistency*	تعمیم *Generalization*
یک فرضیه سازگار است اگر بر روی همه‌ی نمونه‌های آموزشی درست باشد.	یک فرضیه تعمیم‌پذیر است اگر مقادیر خروجی نمونه‌های جدید را به‌درستی پیش‌بینی کند.
بهینه‌سازی به‌دنبال تنظیم یک مدل برای دستیابی به بهترین کارآیی ممکن بر روی داده‌های آموزشی است.	تعمیم مشخص می‌کند که مدل آموزش‌دیده چه‌قدر خوب بر روی داده‌هایی که پیش از این دیده نشده‌اند عمل می‌کند.

بده‌بستان میان **سازگاری ـ تعمیم‌پذیری**:
فرضیه‌های پیچیده با سازگاری کامل و فرضیه‌های ساده‌تر با تعمیم‌پذیری بالاتر

بیش‌برازش و کم‌برازش

OVERFITTING AND UNDERFITTING

بیش‌برازش *Overfitting*	دقت مدل روی داده‌های آموزشی بیشتر از دقت مدل روی داده‌های اعتبارسنجی شده است.
کم‌برازش *Underfitting*	شبکه هنوز همه‌ی الگوهای مربوط در داده‌های آموزشی را مدل نکرده است.

جلوگیری از بیش‌برازش

رگولاریزاسیون

بهترین راه‌حل برای جلوگیری از بیش‌برازش، استفاده از داده‌های بیشتر می‌باشد.
مدلی که بر روی تعداد داده‌ی بیشتری آموزش دیده باشد، طبیعتاً تعمیم‌پذیری بالاتری خواهد داشت.

بهترین راه‌حل بعدی، تعدیل کمیت اطلاعاتی است که مدل مجاز به ذخیره کردن آنهاست،
یا اضافه کردن قیدها بر روی اطلاعاتی است که مدل مجاز به ذخیره کردن آنهاست.

اگر یک شبکه فقط برای حفظ کردن تعداد کوچکی از الگوها توانایی داشته باشد،
فرآیند بهینه‌سازی آن را مجبور خواهد کرد که بر روی برجسته‌ترین الگوها تمرکز کند،
که این شانس بهتری برای تعمیم‌دهی بالاتر دارد.

رگولاریزاسیون *Regularization*	فرآیند مقابله با بیش‌برازش از طریق تعدیل/منظم‌سازی وزن‌ها

تعمیم‌پذیر کردن شبکه‌های عصبی

برای ایجاد تعمیم‌پذیری، روی‌کرد کلی سعی در یافتن ساده‌ترین شبکه است که بر داده‌ها fit شود.

تکنیک‌های رگولاریزاسیون

(منظم‌سازی)

REGULARIZATION TECHNIQUES

کاهش اندازه‌ی شبکه
Reducing the network's size

افزودن رگولاریزاسیون وزنی
Adding weight regularization

افزودن برون‌اندازی Drop-out
Adding dropout

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه

REDUCING THE NETWORK'S SIZE

ساده‌ترین راه برای اجتناب از بیش‌برازش، کاهش اندازه‌ی مدل است:
تعداد پارامترهای قابل یادگیری در مدل (تابعی از تعداد لایه‌ها و تعداد واحدها در هر لایه)

ظرفیت مدل *Model's Capacity*	تعداد پارامترهای قابل یادگیری در مدل

مدلی که پارامترهای بیشتری دارد، **ظرفیت حفظ‌کردن** بیشتری دارد
و بنابراین، به‌سادگی می‌تواند یک نگاشت دیکشنری مانند بین نمونه‌های آموزشی و تارگت آنها را یاد بگیرد،
که یک نگاشت بدون هرگونه قدرت تعمیم است.
(به‌ویژه در مدل‌های عمیق که ظرفیت آنها بالاست)

از طرف دیگر، اگر شبکه منابع حفظ کردن محدودی داشته باشد،
قادر نخواهد بود این نگاشت را به‌سادگی یاد بگیرد؛
پس برای می‌نیمم‌سازی loss به سمت یادگیری بازنمایی‌های فشرده‌ای کشیده می‌شود که
قدرت پیش‌بینی تارگت‌ها را داشته باشند.

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه: تعادل بین ظرفیت بسیار زیاد و ظرفیت ناکافی

مدل مورد استفاده باید تعداد کافی پارامتر داشته باشد تا دچار **کم‌برازش** نشود.
یک وضعیت بده‌بستان (مصالحه) وجود دارد
که باید بین ظرفیت بسیار زیاد و ظرفیت ناکافی پیدا شود.

هیچ فرمول جادویی برای تعیین تعداد درست لایه‌ها یا اندازه‌ی درست هر لایه وجود ندارد.

برای یافتن اندازه‌ی درست مدل، باید آرایه‌ای از معماری‌های مختلف روی مجموعه‌ی اعتبارسنجی ارزیابی شود.

به‌طور کلی، برای یافتن اندازه‌ی مناسب مدل،
از تعداد نسبتاً کمی لایه و پارامتر شروع می‌کنیم
و اندازه‌ی لایه‌ها را اضافه می‌کنیم یا لایه‌های جدیدی اضافه می‌کنیم
تا با توجه به اتلاف روی داده‌های اعتبارسنجی، افت بازدهی را مشاهده کنیم.

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها)

Original model (the movie-review classification network)

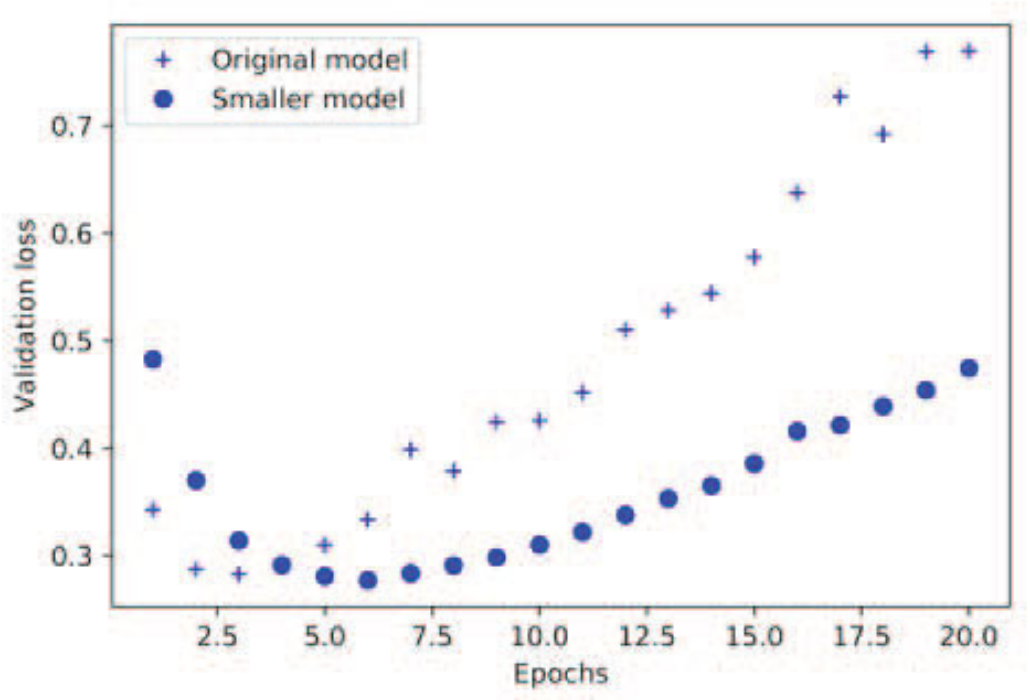
```
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

Version of the model with lower capacity

```
model = models.Sequential()
model.add(layers.Dense(4, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(4, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها) : نمودار اتلاف اعتبارسنجی



Effect of model capacity on validation loss: trying a smaller model

شبکه‌ی کوچک‌تر، دیرتر از شبکه‌ی اصلی شروع به بیش‌برازش می‌کند (پس از ۶ اپک در مقابل ۴ اپک).
و با شروع بیش‌برازش، کارآیی آن کندتر تنزل می‌یابد.

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها) : افزایش ظرفیت شبکه

Original model (the movie-review classification network)

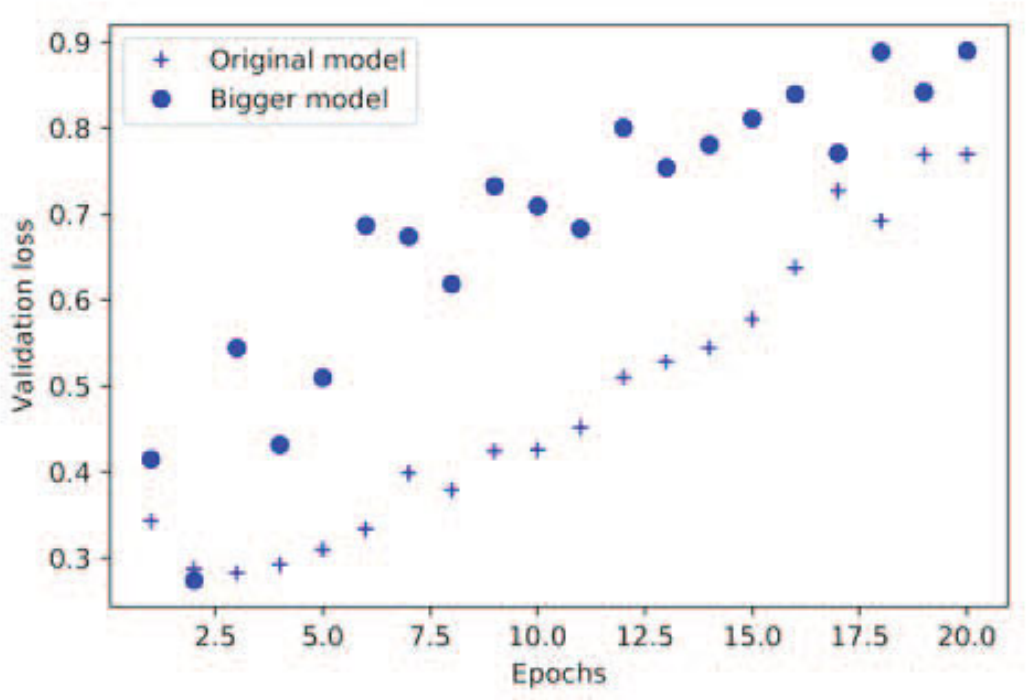
```
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

Version of the model with higher capacity

```
model = models.Sequential()
model.add(layers.Dense(512, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها) : افزایش ظرفیت شبکه : نمودار اتلاف اعتبارسنجی

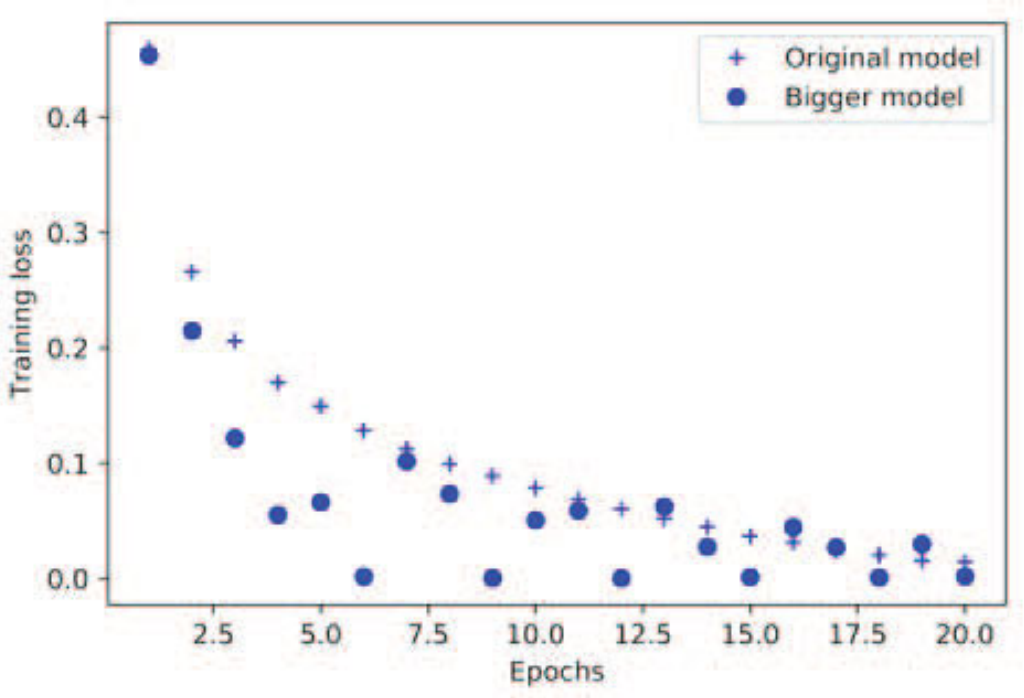


Effect of model capacity on validation loss: trying a bigger model

شبکه‌ی بزرگ‌تر، تقریباً بلافاصله پس از اپک اول شروع به بیش‌برازش می‌کند
و بیش‌برازش آن بسیار شدیدتر است.
سیگنال اتلاف اعتبارسنجی نیز نویزی‌تر است.

تکنیک‌های رگولاریزاسیون

کاهش اندازه‌ی شبکه : مثال (طبقه‌بندی نقد فیلم‌ها) : نمودار اتلاف داده‌های آموزشی



Effect of model capacity on training loss: trying a bigger model

شبکه‌ی بزرگ‌تر در میزان اتلاف داده‌های آموزشی خود سریعاً به نزدیک صفر دست یافته است.
هرچه شبکه ظرفیت بیشتری داشته باشد، داده‌های آموزشی را سریع‌تر می‌تواند مدل کند (منجر به اتلاف کمتر)
اما به بیش‌برازش بیشتر مشکوک می‌شود (در اثر اختلاف بزرگ بین اتلاف‌های آموزشی و اعتبارسنجی).

تکنیک‌های رگولاریزاسیون

اصل تیغه‌ی اوخامی

OCKHAM'S RAZOR

اصل تیغه‌ی اوخامی
Ockham's Razor

ساده‌ترین مدلی که داده‌ها را توضیح می‌دهد، بیابید.

Find the simplest model that explains the data.

هرچه مدل پیچیده‌تر باشد، امکان خطا بیشتر می‌شود.

وقتی دو توضیح برای چیزی داده شده باشد، توضیحی که ساده‌تر است شانس بیشتری برای درست بودن دارد.
(توضیحی که فرضیات کمتری را در نظر می‌گیرد.)

استفاده از این ایده در مدل‌هایی که با شبکه‌های عصبی یاد گرفته می‌شوند:
با داشتن مقداری داده‌های آموزشی و یک معماری شبکه،
چندین مجموعه از مقادیر وزن‌ها (چندین مدل‌ها) میتوانند داده‌ها را توضیح دهند.
مدل‌های ساده‌تر کمتر از مدل‌های پیچیده شانس گرفتار شدن در بیش‌برازش را دارند.

تکنیک‌های رگولاریزاسیون

افزودن رگولاریزاسیون وزنی

ADDING WEIGHT REGULARIZATION

مدل‌های ساده‌تر کمتر از مدل‌های پیچیده شانس گرفتار شدن در بیش‌برازش را دارند.
مدل ساده‌تر در این مضمون، مدلی است که در آن توزیع مقادیر پارامتر آنتروپی کمتری داشته باشد.
(یا مدلی که تعداد پارامتر کمتری داشته باشد.)

پس یک راه متداول برای کاهش بیش‌برازش، قرار دادن قیدهایی بر روی پیچیدگی یک شبکه است:
مجبور کردن وزن‌های شبکه به اینکه تنها مقادیر کوچک را بپذیرند
⇐ توزیع مقادیر وزن‌ها منظم‌تر می‌شود (more regular): **رگولاریزاسیون وزنی**

رگولاریزاسیون وزنی *Weight Regularization*	اضافه کردن یک مقدار هزینه مرتبط با داشتن وزن‌های بزرگ به تابع اتلاف شبکه

انواع متداول رگولاریزاسیون وزنی	○ **رگولاریزاسیون ال۱** (L1 regularization) [the L1 norm of the weights]: هزینه‌ی اضافه شده به اتلاف متناسب با قدرمطلق مقدار ضرایب وزنی است. ○ **رگولاریزاسیون ال۲** (L2 regularization) [the L2 norm of the weights]: هزینه‌ی اضافه شده به اتلاف متناسب با مجذور مقدار ضرایب وزنی است.

Regularization: Add term to loss

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$ (Weight decay)

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Ref: http://cs231n.stanford.edu/

تکنیک‌های رگولاریزاسیون

افزودن رگولاریزاسیون وزنی در کراس

In **Keras**, weight regularization is added
by passing weight regularizer instances to layers as keyword arguments.

Let's add L2 weight regularization to the movie-review classification network.

Adding L2 weight regularization to the model

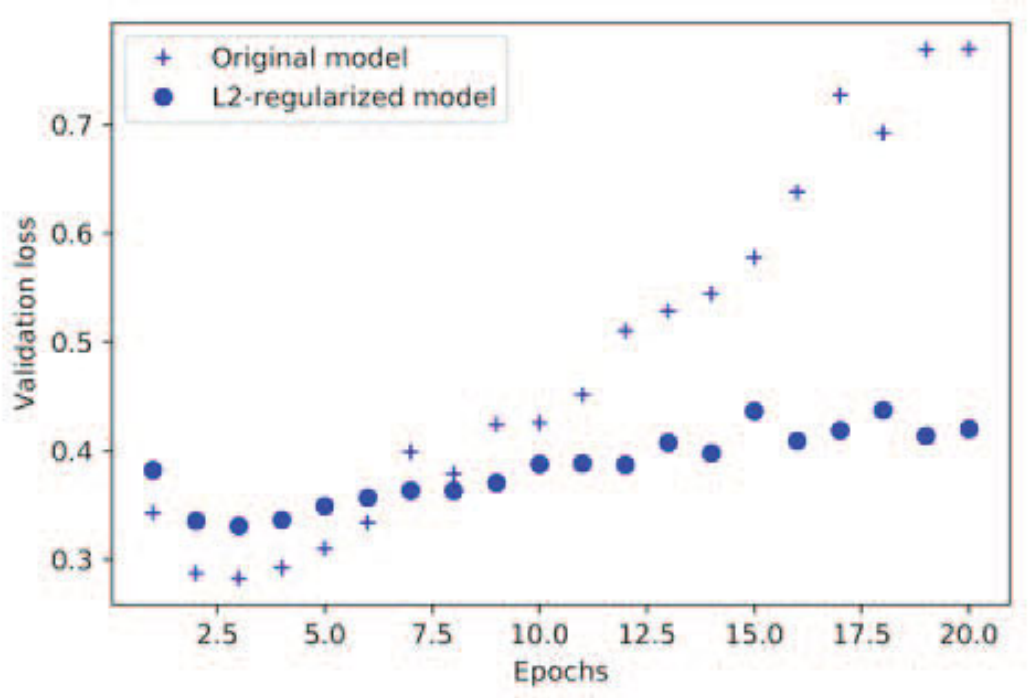
```
from keras import regularizers

model = models.Sequential()
model.add(layers.Dense(16, kernel_regularizer=regularizers.l2(0.001),
                       activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, kernel_regularizer=regularizers.l2(0.001),
                       activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

`l2(0.001)` means every coefficient in the weight matrix of the layer
will add `0.001 * weight_coefficient_value` to the total loss of the network.

تذکر: جریمه‌ی رگولاریزاسیون تنها در زمان آموزش اضافه می‌شود،
برای همین میزان اتلاف شبکه در زمان آموزش بسیار بزرگ‌تر از زمان تست است.

تکنیک‌های رگولاریزاسیون

افزودن رگولاریزاسیون وزنی در کراس: نمودار اتلاف اعتبارسنجی



Effect of L2 weight regularization on validation loss

مدل دارای رگولاریزاسیون L2 نسبت به مدل مرجع در برابر بیش‌برازش بیشتر مقاومت می‌کند.
اگرچه هر دو مدل تعداد یکسانی پارامتر دارند.

تکنیک‌های رگولاریزاسیون

افزودن رگولاریزاسیون وزنی در کراس: گزینه‌ها

As an alternative to L2 regularization,
you can use one of the following **Keras** weight regularizers.

Different weight regularizers available in Keras

```
from keras import regularizers
regularizers.l1(0.001)                  # L1 regularization
regularizers.l1_l2(l1=0.001, l2=0.001) # Simultaneous L1 and L2 regularization
```

تکنیک‌های رگولاریزاسیون

اضافه کردن برون‌اندازی

ADDING DROPOUT

برون‌اندازی/Dropout یکی از مؤثرترین و متداول‌ترین تکنیک‌های رگولاریزاسیون برای شبکه‌های عصبی است.
(توسعه یافته توسط هینتون و دانشجویان او در دانشگاه تورنتو)

برون‌اندازی *Dropout*	اعمال Dropout به یک لایه، عبارت است از حذف تصادفی (صفر کردن) تعدادی از ویژگی‌های خروجی آن لایه در هنگام آموزش

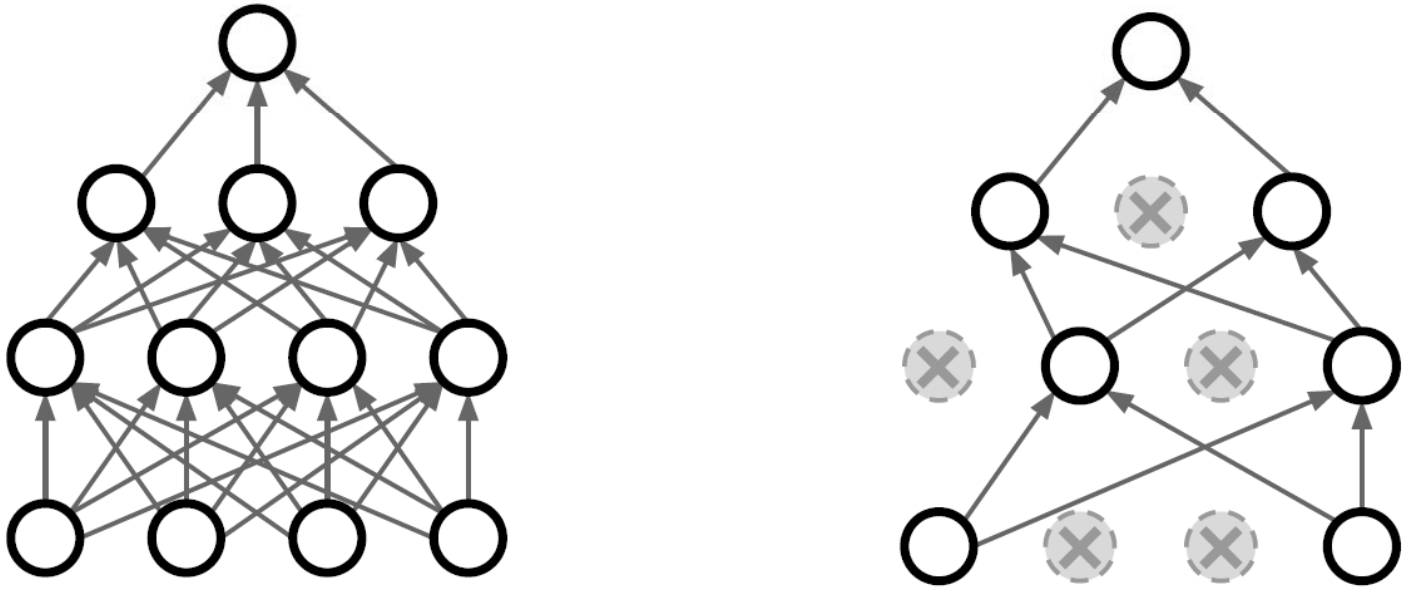
نرخ برون‌اندازی *Dropout Rate*	کسر تعداد ویژگی‌هایی که صفر شده و خارج می‌شوند. این مقدار معمولاً بین 0.2 و 0.5 است.

در زمان تست، هیچ واحدی dropout نمی‌شود؛
در عوض، خروجی لایه‌ها با فاکتوری برابر با نرخ dropout مقیاس می‌شوند.
(برای متعادل‌سازی بر اساس این واقعیت که تعداد واحدهای بیشتری نسبت به زمان آموزش فعال است.)

Regularization: Dropout

In each forward pass, randomly set some neurons to zero

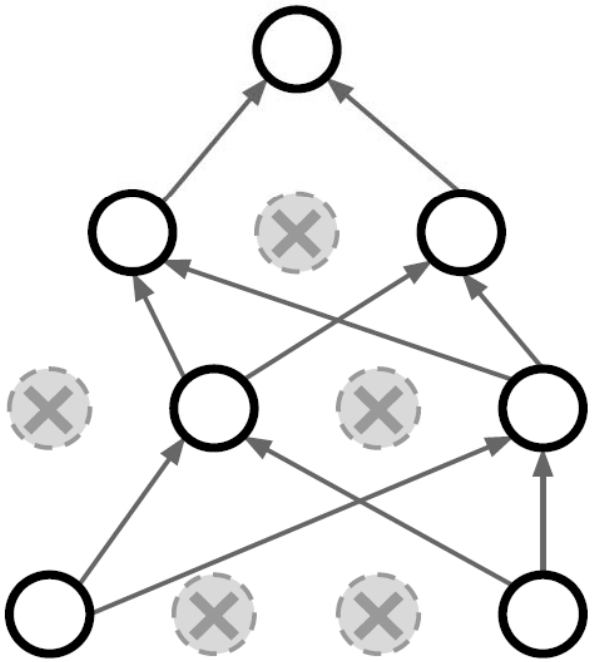
Probability of dropping is a hyperparameter; 0.5 is common



Ref: http://cs231n.stanford.edu/

Regularization: Dropout

How can this possibly be a good idea?

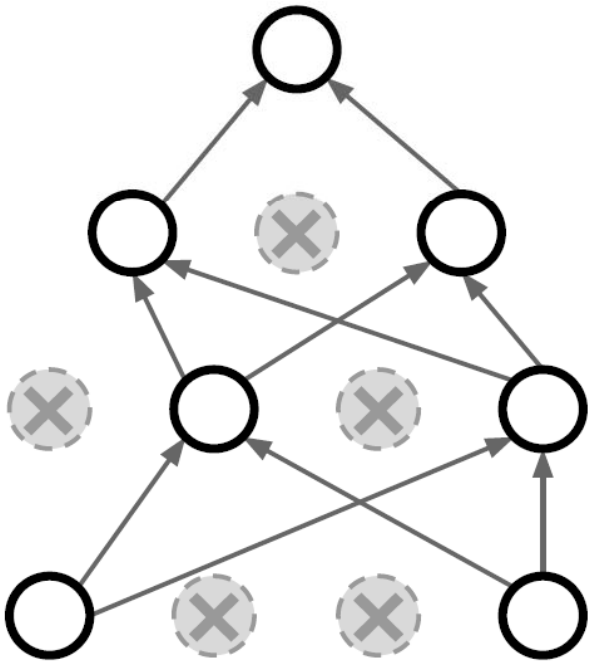


Forces the network to have a redundant representation;
Prevents co-adaptation of features

Ref: http://cs231n.stanford.edu/

Regularization: Dropout

How can this possibly be a good idea?



Another interpretation:

Dropout is training a large **ensemble** of models (that share parameters).

Each binary mask is one model

An FC layer with 4096 units has
$2^{4096} \sim 10^{1233}$ possible masks!
Only $\sim 10^{82}$ atoms in the universe...

Ref: http://cs231n.stanford.edu/

Dropout: Test time

Dropout makes our output random!

Output (label) Input (image)

$$y = f_W(x, z)$$

Random mask

Want to "average out" the randomness at test-time

$$y = f(x) = E_z\big[f(x, z)\big] = \int p(z) f(x, z) dz$$

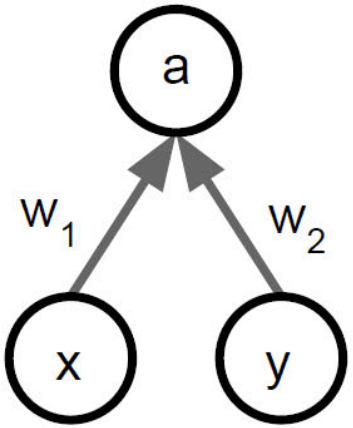
But this integral seems hard …

Ref: http://cs231n.stanford.edu/

Dropout: Test time

Want to approximate the integral

$$y = f(x) = E_z\big[f(x,z)\big] = \int p(z) f(x,z) dz$$



Consider a single neuron.

At test time we have: $E\big[a\big] = w_1 x + w_2 y$

During training we have:

$$\begin{aligned} E[a] =& \frac{1}{4}(w_1 x + w_2 y) + \frac{1}{4}(w_1 x + 0y) \\ &+ \frac{1}{4}(0x + 0y) + \frac{1}{4}(0x + w_2 y) \\ =& \frac{1}{2}(w_1 x + w_2 y) \end{aligned}$$

At test time, **multiply** by dropout probability

Ref: http://cs231n.stanford.edu/

Dropout: Test time

```
def predict(X):
  # ensembled forward pass
  H1 = np.maximum(0, np.dot(W1, X) + b1) * p # NOTE: scale the activations
  H2 = np.maximum(0, np.dot(W2, H1) + b2) * p # NOTE: scale the activations
  out = np.dot(W3, H2) + b3
```

At test time all neurons are active always
=> We must scale the activations so that for each neuron:
output at test time = expected output at training time

Ref: http://cs231n.stanford.edu/

تکنیک‌های رگولاریزاسیون

اضافه کردن برون‌اندازی

Consider a Numpy matrix containing the output of a layer, layer_output,
of shape `(batch_size, features)`.
At training time, we zero out at random a fraction of the values in the matrix:

```
layer_output *= np.random.randint(0, high=2, size=layer_output.shape)
                        # At training time, drops out 50% of the units in the output
```

At test time, we scale down the output by the dropout rate. Here, we scale by 0.5
(because we previously dropped half the units):

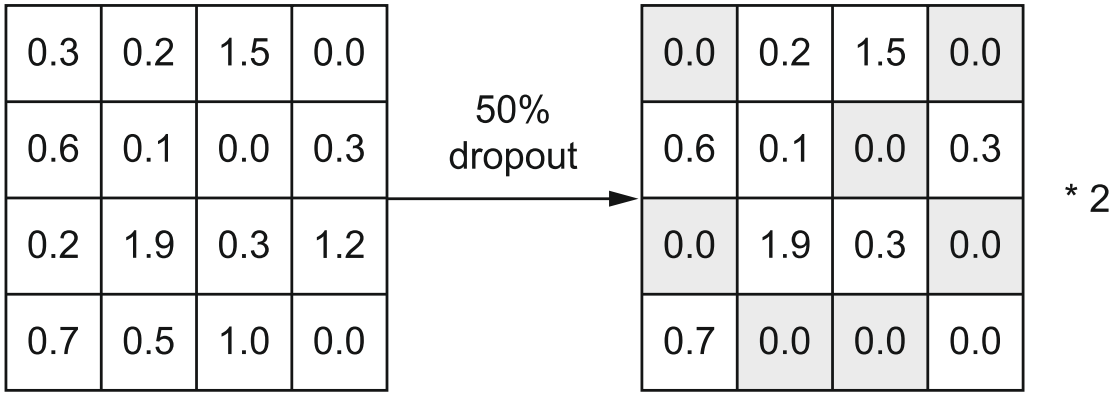
```
layer_output *= 0.5                              # At test time
```

تکنیک‌های رگولاریزاسیون

اضافه کردن برون‌اندازی

Note that this process can be implemented by doing both operations at training time and leaving the output unchanged at test time, which is often the way it's implemented in practice:

```
layer_output *= np.random.randint(0, high=2, size=layer_output.shape)
                                                    # At training time
layer_output /= 0.5
                    # Note that we're scaling up rather scaling down in this case.
```



Dropout applied to an activation matrix at training time, with rescaling happening during training. At test time, the activation matrix is unchanged.

تکنیک‌های رگولاریزاسیون

اضافه کردن برون‌اندازی: منشأ ایده

تکنیک برون‌اندازی/ Dropout ممکن است عجیب و دلبخواهی به‌نظر برسید.
چرا این تکنیک بیش‌برازش را کاهش می‌دهد؟
هینتون بیان می‌کند که این روش را
غیر از سایر موارد، از یک مکانیسم پیشگیری از کلاهبرداری در بانک‌ها ایده گرفته است:

«من به بانک رفته بودم، تحویل‌دارها جای خود را با یکدیگر عوض کرده بودند و من دلیل این کار را از یکی از آنها پرسیدم. او گفت که نمی‌داند، اما آنها زیاد جای خود را تغییر می‌دهند.
من با خودم فکر کردم که دلیل آن حتماً این است که پیشگیری موفقیت‌آمیز از کلاهبرداری از بانک نیازمند همکاری بین کارمندان است. این نکته باعث شد تا متوجه بشوم تغییر مکان تصادفی یک زیرمجموعه‌ی متفاوت از نرون‌ها در هر مثال می‌تواند مانع از توطئه و دسیسه شود و بنابراین بیش‌برازش را کاهش می‌دهد.»

ایده و مفهوم اصلی:

وارد کردن نویز در مقادیر خروجی یک لایه،
می‌تواند سبب شود الگوهای اتفاقی که چشمگیر نیستند (توطئه‌ها) بشکنند
اما اگر هیچ تداخل و نویزی وجود نداشته باشد، شبکه شروع به حفظ کردن داده‌ها می‌کند (بیش‌برازش).

تکنیک‌های رگولاریزاسیون

اضافه کردن Dropout در کراس

In **Keras**, you can introduce dropout in a network via the Dropout layer, which is applied to the output of the layer right before it

```
model.add(layers.Dropout(0.5))
```

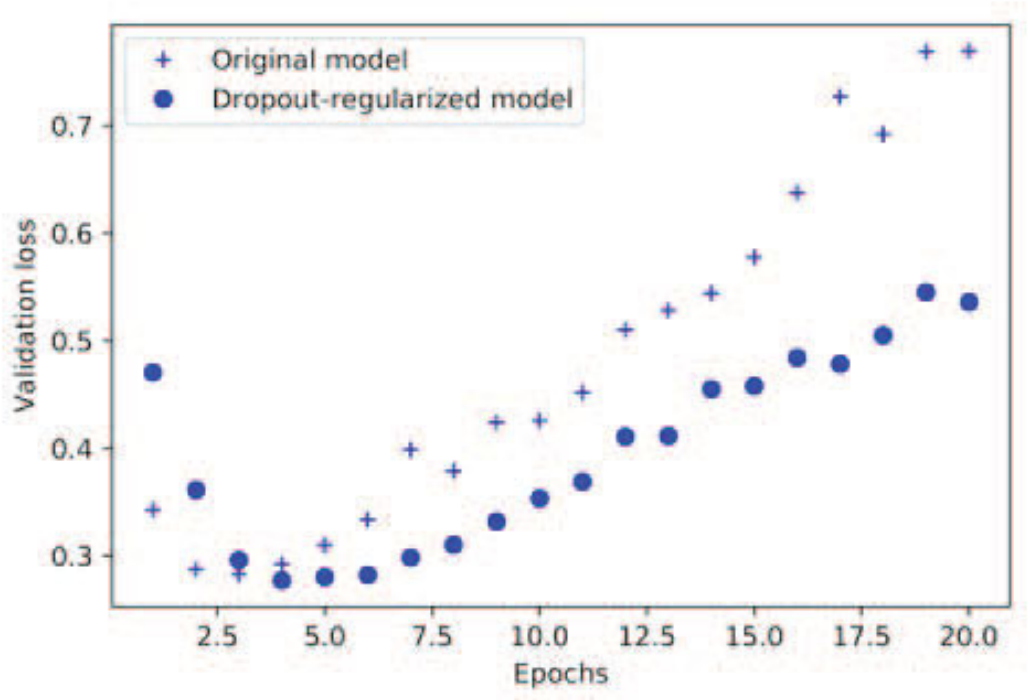
Let's add two Dropout layers in the IMDB network to see how well they do at reducing overfitting.

Adding dropout to the IMDB network

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

تکنیک‌های رگولاریزاسیون

اضافه کردن Dropout : نمودار اعتبارسنجی



Effect of dropout on validation loss

این نمودار بیانگر پیشرفت آشکاری نسبت به شبکه‌ی مرجع است.

جمع‌بندی: اجتناب از بیش‌برازش در شبکه‌های عصبی

RECAP: TO PREVENT OVERFITTING IN NEURAL NETWORKS

متداول‌ترین روش‌ها برای جلوگیری از بروز بیش‌برازش در شبکه‌های عصبی:

یادگیری عمیق

مبانی یادگیری ماشینی

۵

گردش کار جامع در یادگیری ماشینی

گردش کار جامع در یادگیری ماشینی

The Universal Workflow of Machine Learning

گردش کار جامع در یادگیری ماشینی

تعریف مسئله و گردآوری مجموعه‌داده

DEFINING THE PROBLEM AND ASSEMBLING A DATASET

۱ تعریف مسئله و گردآوری مجموعه‌داده
Defining the problem and assembling a dataset

تعریف مسئله

- داده‌های ورودی؟ چه چیزی باید پیش‌بینی شود؟
- ورودی؟ خروجی؟
- نوع مسئله؟ (طبقه‌بندی دودویی، رگرسیون، خوشه‌بندی، تولید، ...)

شناسایی نوع مسئله، در انتخاب مدل، تابع اتلاف و ... ما را راهنمایی می‌کند.

فرضیات اساسی

- فرض می‌شود که می‌توان خروجی‌ها را بر اساس ورودی‌های آنها پیش‌بینی کرد. [وجود رابطه‌ی تابعی بین ورودی‌ها و خروجی‌ها]
- فرض می‌شود که داده‌های موجود حاوی اطلاعات کافی برای یادگیری رابطه‌ی بین ورودی‌ها و خروجی‌ها هستند. [میزان داده‌ی کافی]

گردش کار جامع در یادگیری ماشینی

تعریف مسئله و گردآوری مجموعه‌داده: مسائل حل‌ناپذیر

UNSOLVABLE PROBLEMS

مسائل غیرایستان
Nonstationary Problems

مسائلی که هدف از آنها کشف تابعی است که ضابطه‌ی آن با زمان تغییر می‌کند.

مثال: مسئله‌ی سیستم توصیه‌گر لباس
توصیه متناسب با زمان سال متغیر است،
پس داده‌های آموزشی باید حاوی ویژگی زمان سال باشند.

گردش کار جامع در یادگیری ماشینی

تعریف مسئله و گردآوری مجموعه‌داده : محدودیت‌ها

یادگیری ماشینی تنها می‌تواند
برای حفظ کردن الگوهایی که در مجموعه داده‌های آموزشی حاضر است
استفاده شود.

«چیزی را می‌توانیم بازشناسی کنیم که قبلاً آن را دیده باشیم»

استفاده از یادگیری ماشینی آموزش‌دیده بر روی داده‌های گذشته برای پیش‌بینی آینده،
این فرض را در خود دارد که:
«آینده مانند گذشته رفتار می‌کند»
که خیلی اوقات برقرار نیست!

گردش کار جامع در یادگیری ماشینی

انتخاب یک معیار موفقیت

CHOOSING A MEASURE OF SUCCESS

۲ انتخاب یک معیار موفقیت
Choosing a measure of success

برای اینکه بتوان چیزی را کنترل کرد، باید بتوان آن را مشاهده کرد.
برای دستیابی به موفقیت باید آن را تعریف کنیم: دقت؟ یادآوری؟ نرخ حفظ مشتری؟ ...

معیار موفقیت، راهنمایی برای انتخاب یک تابع هزینه خواهد بود.

- برای مسائل طبقه‌بندی متوازن (احتمال کلاس‌ها مساوی)، **صحت (Accuracy)** و **سطح زیر منحنی ROC** یک معیار متداول است (ROC AUC).
- برای مسائل طبقه‌بندی نامتوازن (احتمال کلاس‌ها نامساوی)، **دقت (Precision)** و **یادآوری (Recall)** متداول است.
- برای مسائل رتبه‌بندی یا طبقه‌بندی چندبرچسبی، **میانگین دقت متوسط (MAP)** متداول است.

تعریف معیار موفقیت اختصاصی برای کاربردهای گوناگون، متداول است.

گردش کار جامع در یادگیری ماشینی

تصمیم‌گیری در مورد پروتکل ارزیابی

DECIDING ON AN EVALUATION PROTOCOL

۳ تصمیم‌گیری در مورد پروتکل ارزیابی
Deciding on an evaluation protocol

وقتی که هدف مشخص شد، باید شیوه‌ای برای سنجش **پیشرفت فرآیند یادگیری** مشخص شود.

سه پروتکل متداول برای ارزیابی داریم:

اعتبارسنجی ساده نگه‌داشت–برون‌گذاشت *Simple Hold-Out Validation*	روش مناسب برای زمانی که حجم فراوانی داده در اختیار داریم. (در بیشتر موارد این روش به‌اندازه‌ی کافی خوب است)
اعتبارسنجی *K-fold* *K-fold Validation*	انتخاب درست برای مواقعی که نمونه‌های بسیار اندکی در اختیار داریم و روش ساده اطمینان‌بخش نیست.
اعتبارسنجی *K-fold* تکراری با بُر زدن *Iterated K-fold Validation with Shuffling*	برای اجرای ارزیابی بسیار دقیق مدل در زمانی که داده‌های کمی در دسترس است.

گردش کار جامع در یادگیری ماشینی

آماده‌سازی داده‌ها

PREPARING THE DATA

۴	آماده‌سازی داده‌ها *Preparing the data*

ابتدا باید داده‌ها را به‌گونه‌ای که بتوانند وارد یک مدل یادگیری ماشینی (در اینجا شبکه‌ی عصبی) شوند، فرمت‌بندی کنید.

فرمت‌بندی برای آماده‌سازی داده‌ها

- داده‌ها باید در قالب تانسورها فرمت‌بندی شوند.
- مقادیر موجود در تانسورها باید در قالب اعداد کوچک مقیاس شوند. (بازه‌ی [0,1] یا [-1,1])
- نرمال‌سازی مستقل ویژگی‌ها برای داده‌های ناهمگن
- انجام مهندسی ویژگی‌ها به‌خصوص برای مسائلی با داده‌های کوچک

زمانی که تانسورهای داده‌های ورودی و تارگت آماده شدند، می‌توانیم آموزش مدل را شروع کنیم.

گردش کار جامع در یادگیری ماشینی

توسعه‌ی یک مدل که بهتر از یک مدل پایه عمل کند

DEVELOPING A MODEL THAT DOES BETTER THAN A BASELINE

۵ توسعه‌ی یک مدل که بهتر از یک مدل پایه عمل کند
Developing a model that does better than a baseline

هدف این مرحله، توسعه‌ی مدلی است که به **قدرت آماری** دست پیدا کند.

قدرت آماری
Statistical Power

مدلی دارای قدرت آماری است که
قادر به شکست دادن یک مدل پایه‌ی تصادفی باشد.

برای مثال: مدل پایه تصادفی برای مسئله‌ی طبقه‌بندی دو کلاسی دارای دقت $0.5 = ½$ است.
مدل پایه تصادفی برای مسئله‌ی طبقه‌بندی n–کلاسی دارای دقت $1/n$ است.

دستیابی به مدلی با قدرت آماری، همیشه ممکن نیست.
اگر پس از آزمایش چندین معماری مستدل نتوانیم مدل پایه‌ی تصادفی را شکست دهیم،
ممکن است به این دلیل باشد که پاسخ پرسشی که به‌دنبال آن هستیم در داده‌های ورودی وجود ندارد.
یا ممکن است فرضیات اساسی (وجود رابطه‌ی تابعی / داده‌ی کافی) نادرست باشند.

گردش کار جامع در یادگیری ماشینی

توسعه‌ی یک مدل که بهتر از یک مدل پایه عمل کند: پارامترها

با فرض درست بودن فرضیات اولیه،
باید در مورد سه انتخاب کلیدی برای ساختن اولین مدل کاری خود تصمیم‌گیری کنید:

سه انتخاب کلیدی برای ساختن اولین مدل کاری

- تابع فعالیت لایه‌ی آخر [بر اساس نوع خروجی شبکه]
- تابع اتلاف [لزوم مطابقت با مسئله‌ی مورد نظر]
- پیکربندی بهینه‌ساز (الگوریتم بهینه‌سازی، نرخ یادگیری آن، ...)

با توجه به انتخاب تابع اتلاف، همواره امکان بهینه‌سازی مستقیم برای معیارهای موفقیت مسئله وجود ندارد.
در برخی موارد هیچ راه آسانی برای تبدیل یک معیار موفقیت به یک تابع هزینه وجود ندارد.
تابع اتلاف باید برای یک دسته‌ی کوچک از داده‌ها قابل محاسبه باشد.
تابع اتلاف باید حتی برای یک نقطه هم قابل محاسبه باشد.
تابع اتلاف باید مشتق‌پذیر باشد تا بتوان از آن در الگوریتم پس‌انتشار خطا استفاده کرد.
[برای مثال معیار ROC AUC مشتق‌پذیر نیست
و برای همین در مسائل طبقه‌بندی از معیارهای نماینده‌ی آن مانند آنتروپی متقابل استفاده می‌شود.]

گردش کار جامع در یادگیری ماشینی

توسعه‌ی یک مدل که بهتر از یک مدل پایه عمل کند: انتخاب تابع فعالیت و تابع اتلاف برای مسائل گوناگون

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

گردش کار جامع در یادگیری ماشینی

افزایش مقیاس: توسعه‌ی یک مدل که بیش‌برازش کند

SCALING UP: DEVELOPING A MODEL THAT OVERFITS

۶ افزایش مقیاس: توسعه‌ی یک مدل که بیش‌برازش کند
Scaling up: developing a model that overfits

وقتی به مدلی رسیدیم که دارای قدرت آماری است، این پرسش مطرح می‌شود که:
آیا این مدل به‌اندازه‌ی کافی قدرت‌مند است؟
[تعداد لایه‌ها و واحدها کافی است؟ ...]

کشمکش اساسی در یادگیری ماشینی بین سازگاری و تعمیم‌دهی است.
مدل ایده‌آل، مدلی است که درست در مرز میان **کم‌برازش** و **بیش‌برازش** قرار بگیرد:
مرز میان کم‌ظرفیتی و بیش‌ظرفیتی؛
برای پیدا کردن این مرز ابتدا باید از آن عبور کنیم.

برای تعیین بزرگی مدل مورد نیاز، باید مدلی ایجاد کرد که دچار بیش‌برازش شود:

- لایه‌ها را اضافه می‌کنیم.
- لایه‌ها را بزرگ‌تر می‌کنیم.
- فرآیند آموزش را طی اپک‌های بیشتری تکرار می‌کنیم.

اتلاف روی داده‌های آموزشی و اعتبارسنجی را تحت نظارت قرار می‌دهیم.
هنگامی که کارآیی مدل روی داده‌های اعتبارسنجی شروع به افت کرد، به بیش‌برازش رسیده‌ایم.

گردش کار جامع در یادگیری ماشینی

رگولاریزاسیون مدل و تنظیم هایپرپارامترها

REGULARIZING THE MODEL AND TUNING YOUR HYPERPARAMETERS

۷ رگولاریزاسیون مدل و تنظیم هایپرپارامترها
Regularizing the model and tuning your hyperparameters

این مرحله بیشترین زمان را صرف می‌کند:
مدل باید به‌طور مکرر اصلاح شود، تحت آموزش مجدد قرار گیرد و روی داده‌های اعتبارسنجی ارزیابی شود.
مجدداً مدل اصلاح می‌شود و تکرار می‌شود تا زمانی که مدل به بهترین حالت برسد.

موارد قابل امتحان کردن

- اضافه کردن برون‌اندازی / Dropout
- آزمون معماری‌های مختلف: اضافه / حذف لایه‌ها
- اضافه کردن رگولاریزاسیون وزنی
- آزمون هاپیرپارامترهای مختلف (تعداد واحد هر لایه / نرخ یادگیری بهینه‌ساز، ...)
- مهندسی ویژگی‌ها: اضافه کردن ویژگی جدید، حذف ویژگی غیرمفید

نکته‌ی مهم: هر زمان که فیدبک فرآیند اعتبارسنجی برای تنظیم مدل استفاده می‌شود،
مقداری نشت اطلاعات در مورد فرآیند اعتبارسنجی به درون مدل داریم.
اگر تنها چند بار این اتفاق رخ دهد، خطری ندارد،
اما اگر به‌طور منظم و به‌دفعات رخ بدهد، سرانجام منجر به بیش‌برازش مدل نسبت به فرآیند اعتبارسنجی می‌شود.
در نتیجه فرآیند ارزیابی، قابلیت اطمینان کمتری خواهد داشت.

گردش کار جامع در یادگیری ماشینی

پایان کار

وقتی به یک پیکربندی راضی‌کننده برای مدل رسیدیم،
مدل نهایی را روی تمام داده‌های موجود (آموزشی و اعتبارسنجی) مورد آموزش قرار می‌دهیم
و آن را برای آخرین بار روی مجموعه‌ی آزمایشی ارزیابی می‌کنیم.

اگر مشخص شود که کارآیی در مجموعه‌ی آزمایشی خیلی بدتر از کارآیی در مجموعه‌ی آموزشی است،
ممکن است به این معنا باشد که روال اعتبارسنجی مورد استفاده قابل اطمینان نبوده است،
یا در هنگام تنظیم پارامترهای مدل شروع به بیش‌برازش نسبت به داده‌های اعتبارسنجی کرده است.
در این حالت باید به سراغ یک پروتکل ارزیابی با قابلیت اطمینان بالاتر برویم.

مبانی یادگیری ماشینی

خلاصه

- Define the problem at hand and the data on which you'll train.
  Collect this data, or annotate it with labels if need be.
- Choose how you'll measure success on your problem.
  Which metrics will you monitor on your validation data?
- Determine your evaluation protocol: hold-out validation? K-fold validation?
  Which portion of the data should you use for validation?
- Develop a first model that does better than a basic baseline:
  a model with statistical power.
- Develop a model that overfits.
- Regularize your model and tune its hyperparameters, based on performance on the validation data. A lot of machine-learning research tends to focus only on this step—but keep the big picture in mind.

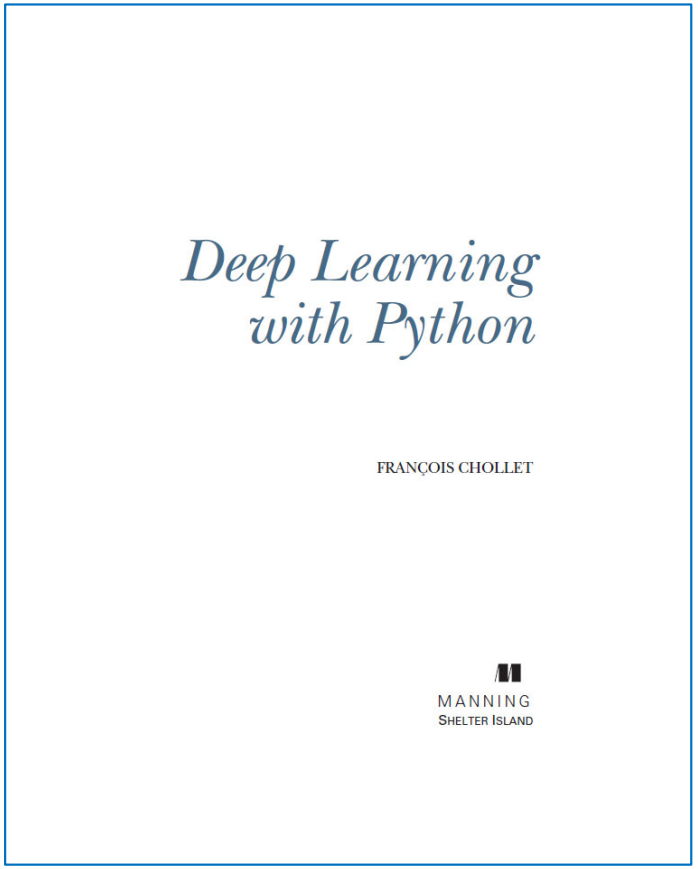
یادگیری عمیق

مبانی یادگیری ماشینی

۶

منابع

منبع اصلی



François Chollet,
Deep Learning with Python,
Manning Publications, 2018.

Chapter 4